Benchmarking computers for seismic processing and imaging

Evgeny Kurin
GEOLAB Ltd.
ekurin@geo-lab.ru
Outline

• O&G HPC status and trends
• Benchmarking: goals and tools
• GeoBenchmark: modules vs. subsystems
• Basic tests with examples
• Tests in action
• Conclusions and directions
O&G HPC status

• Seismic processing and imaging still demand huge computing power

• Significant diversity in computer system characteristics – no “standard computer”

• I/O is still a challenge

• Production applications are more and more complex
Goals of benchmarking

• For vendors
  – Test and promote new hardware

• For end users
  – Evaluate hardware before deployment
  – Locate bottlenecks in existing configurations

• For developers
  – Understand hardware capabilities for algorithm and software design
How benchmarking is done

• Problem-domain benchmarks
  – SPEC (including SPECseis2002*)

• Synthetic benchmarks
  – STREAM, IOZone, Netperf,…

• Target applications and modules
  – SeisSpace, GeoCluster, GeoDepth, …

* Originally developed by C.Mosher and S.Hassanzadeh at ARCO
Why one more benchmark?

• Problem-domain benchmarks
  – Mostly good in system “ranking”
  – Too complex for locating bottlenecks

• Synthetic benchmarks
  – Sometimes not realistic enough

• Target applications
  – Not always available at the time of testing
  – Subject to continuous upgrading
GeoBenchmark

• A set of essentially simple test programs
• Each test represents a class of seismic processing modules
• Each test is targeted at a specific subsystem of a computer
• Tests are reasonably portable
Classes of processing modules

• Single-channel: 1 trace – 1 trace
  – AGC, band-pass filtering, spiking decon, …

• Multi-channel: 1,000 traces – 1 trace
  – Radon transform, post-stack migration, …

• Total access: 1,000,000 traces – 1 trace
  – Pre-stack migration, SRMP, …

• Disk I/O
Computer schematic

Cluster

Node

Cluster interconnect

Node
Node
Node

Shared
storage
system

CPU
Cache

CPU
Cache

CPU
Cache

Main memory

Disk memory
Single-channel modules

- 1 trace to 1 trace
- Data and code fit into the CPU cache
- Many operations on a small portion of data
- “Pure” CPU performance is mostly important for the overall performance
- Suggested test: band-pass filtering in the frequency domain
Test 1: Single-channel

Band-pass filter in the frequency domain

Trace length: 1,000 – 20,000 samples
Data size: ~ 10 – 1000 KB
Number of operations: > 10,000

Algorithm

Compute a filter:
\[ F(\omega) = f(\omega_1, \omega_2) \]

For each trace \( s_i(t) \)
\[ s_i(t) \xrightarrow{FFT} S(\omega) \]
\[ R(\omega) = S(\omega) \times F(\omega) \]
\[ R(\omega) \xrightarrow{FFT^{-1}} r_i(t) \]
Test 1: Example

![Graph showing computation time versus trace length and main memory accessed with different processor speeds: Opteron 2.4GHz, Xeon 2.8GHz, Pentium 3.0GHz.](http://www.geo-lab.ru)
Multi-channel modules

- 1000 traces to 1 trace
- Data and code don’t fit into the CPU cache
- CPU performance and memory bandwidth are equally important for the overall performance
- Suggested test: convolutions of randomly-selected traces in memory
Test 2: Multi-channel

Pair-wise convolutions of traces in a gather

Algorithm

M – traces in a gather
N – samples in a trace
Do $M$ times:
  Randomly select $i$
  Randomly select $j$
  Convolve traces $i$ and $j$

Number of traces: 10 – 10,000
Trace length: 1,000 – 20,000 samples
Data size: ~ 1 – 1000 MB
Test 2: Example
Test 3: Multi-channel on SMP

Kirchhoff post-stack migration

Data size: ~ 100 – 1000 MB
Number of CPUs: 2 - 32
Post-stack or common-offset

\[ D(x_0, z_0) \sim \sum_{x \in A} U(x, t) \times \delta(t - \tau(x_0, z_0, x)) \]
Test 3: Parallelization scheme

- POSIX threads
- Threads do not interact
- Shared variables are read-only
- No special synchronization
Test 3: Example

IBM p570

SGI Altix 330

Performance

Scalability
Total-access modules

• All traces to 1 trace
• Data can occupy all available memory
• CPUs’ performance, memory bandwidth and its accessing mechanism are important for the overall performance
• Suggested test: 2D surface-related multiple prediction (SRMP)
Test 4: Total-access

Surface-related multiple prediction (SRMP) 2D

Algorithm

\[ M(S, R, t) = \sum_{Z \in A} D(S, Z, t) \times \tilde{D}(Z, R, t) \]

Data size: \(~ 1 – 64 \) GB
Number of CPUs: 2 - 32

GEOLAB Ltd.
http://www.geo-lab.ru
Tests 3 vs. 4: vSMP-computer

Test 3 (migration)  
Performance  
![Graph showing performance improvement with increasing number of computing threads]

Test 4 (SRMP)  
Performance  
![Graph showing performance improvement with increasing number of computing threads]

Scalability  
![Graph showing scalability increase with increasing number of computing threads]

![Graph showing scalability increase with increasing number of computing threads]
I/O modules

• A part of any processing sequence
• Often a bottleneck
• Suggested tests
  • Sequential reading
  • Sequential writing
  • Strided (sort-like) reading
  • Reading by a given pattern
Test 5: I/O

Cluster interconnect

Node -> Node -> Node

Shared storage system

Successive read/write

Read every n-th trace (sort)

Read by algorithm-specific pattern
I/O pattern example

3D SRMP: trace access distribution

SP-sorted data

CDP-sorted data

Trace access order

GEOLAB Ltd.  http://www.geo-lab.ru
Test 5: Example, one host

Sequential reading

Strided reading

File size: twice the RAM size
NAS: over Gigabit Ethernet
Trace size: 8k
Stride: 200

GEOLAB Ltd. http://www.geo-lab.ru
Test 5: Example, high-end cluster

SKIF cluster at Moscow State University (#54 in Top500 Nov-2008) with the Panasas ActiveScale storage

GEOLAB Ltd. http://www.geo-lab.ru
In action, 1: code vs. compiler

• GeoBenchmark as a “model” for achieving the best performance with Intel compilers

• Recommendations for preparing the codes for optimal vectorizing and reducing CPI

• Presented at IDF 2008 in Shanghai

He Wanqing, Qiao Nan, Justin Chen, Jin Jun, Wang Zhe, Mao Xiaowei, “Fast Application Characterization Based Performance Tuning and HPC Hardware Solution Design”, SVRS005
Code vs. compiler, analysis

GeoBenchmark* Tuning Radar

GeoBenchmark* is a simple but effective benchmark developed by Evgeny Kurin, Russia as http://geocomputing.narod.ru/benchmark.html

Seismic Benchmark (GeoBenchmark)
Performance Analysis

- Branch Misprediction
- Performance Impact
- L1 Data Cache Miss Rate
- L2 Cache Demand Miss Rate
- Bus Utilization
- TLB miss penalty
- Disk I/O granularity
- Floating Point Instructions Ratio
- Vectorization ratio

GeoBenchmark Unoptimized
GeoBenchmark Optimized

From proceedings of IDF 2008, Shanghai. Courtesy of He Wanqing, Intel Corp.
Code vs. compiler, results

Vectorization: Modify the Source Code

- Simple code may not lead to good performance.
- Evaluate the result of float multiply/divide add to integer variable.
- Break complex heavy loop into multiple simple loops, which can be fully vectorized to use SSE/SSE2/SSE3 instructions.
- Introduce temporary variable to simplify micro-operation.
- Vectorized float multiply add operation.
- Vectorized float to integer conversion operation.
- Vectorized float evaluation operation.
- Near 8 times performance improvement of Quad-Core Intel® Xeon® processor (Clovertown).

From proceedings of IDF 2008, Shanghai. Courtesy of He Wanqing, Intel Corp.
In action, 2: a different platform

Cell Broadband Engine

Original code

```c
float * d = data + iy * nt;
y = yo + dy * iy;
for (iz = 0; iz < nz; iz++)
{
    z = zo + dz * iz;
    hs = (x - y) / velhalf;
    t = sqrt(z*z+hs*hs);
    ftt[iz] = 0.5+t/dt;
}
for (iz = 0; iz < nz; iz++)
{
    itt[iz] = ftt[iz];
}
for (iz = 0; iz < nz; iz++)
{
    if (itt[iz] < nt)
    {
        mod[i] += d[itt[iz]];
    }
}
```

Cell/BE SPE code

```c
d = (float *)diob[b];
y = spu_madd(dy, spu_splats((float)iy), y0);
hs2 = spu_mul(spu_sub(x, y), velhalf_1);
hs2 = spu_mul(hs2, hs2);
qz = qz0;
for (iz = 0; iz < QUARTER_SIZE; iz++)
{
    t = sqrtf4(spu_madd(qz, qz, hs2));
    fixs = spu_madd(t, dt_1, one_half);
    iixs = spu_convts(fixs, 0);
    ix0 = spu_extract(iixs, 0);
    ix1 = spu_extract(iixs, 1);
    ix2 = spu_extract(iixs, 2);
    ix3 = spu_extract(iixs, 3);
    dadd = spu_splats((float)0);
    if (ix0 < nt) dadd = spu_insert(d[ix0], dadd, 0);
    if (ix1 < nt) dadd = spu_insert(d[ix1], dadd, 1);
    if (ix2 < nt) dadd = spu_insert(d[ix2], dadd, 2);
    if (ix3 < nt) dadd = spu_insert(d[ix3], dadd, 3);
    mod[iz] += dadd;
    qz += qdz;
}
```

Test 3: Parallelized post-stack migration

GEOLAB Ltd.
Intel Xeon 5335 vs. Cell/BE

Studying the performance of Cell/BE

Performance

Scalability

Test 4: Surface-related multiple prediction (SRMP)
In action, 3: education & training

• Students
  – Basics of seismic parallel processing

• Developers
  – Performance characteristics of the hardware

• System engineers
  – Functioning of a seismic processing center and possible performance bottlenecks
Conclusions

• A set of simple tests for benchmarking computers used in seismic processing and imaging is suggested

• Each test represents a class of processing algorithms and in the same time corresponds to a computer subsystem

• Tests will complement existing benchmarks for a number of purposes
Directions

• Add new tests
  – WE migration for SMP
  – Reverse time migration for SMP
  – MPI tests

• Clean-up
  – Unifying calling scripts
  – Documenting
  – Implementing immediate graphing
Thanks to

• Dmitri Mosyakov (GeoPrime, Russia)
• Dmitri Lokshtanov (StatoilHydro, Norway)
• Sergey Zhumaitiy (Moscow State University, Russia)
• Dmitri Tkachev (T-Platforms, Russia)
• Ekaterina Kurina (Geological Institute, Russian Academy of Sciences)
• He Wanqing (Intel Corp., China)
• Everyone (GEOLAB Ltd., Russia)
• Organizers of the Oil and Gas HPC Workshop
Resources

• Source codes of GeoBenchmark:

• E-mail:
  – ekurin@geo-lab.ru